

OCR

Oxford Cambridge and RSA

An OCR endorsed
teaching and learning tool

OCR A Level

Computer
Science

H446 – Paper 1



Stacks

Unit 7

Data structures



PG ONLINE

Objectives

- Be familiar with the concept and uses of a stack
- Be able to describe the creation and maintenance of data within a stack
- Be able to describe and apply the following operations: push, pop, peek (or top), test for empty stack, test for full stack
- Be able to explain how a stack frame is used with subroutine calls to store return addresses, parameters and local variables

Abstraction

- Three abstract data types (ADTs) covered so far are a **queue**, **list**, and **linked list**
 - How are the concepts of data hiding and encapsulation used in the implementation of an abstract data type?

Stack definition

- Think of a stack of textbooks
 - The teacher adds to the top of the stack
 - The students removes from the top of the stack
 - Last In First Out = LIFO



Using a stack

- Many examples of stacks occur in everyday life – can you think of examples?
 - Can you think of any examples related to computing?
 - What operations are needed to implement a stack?

Modelling a stack

- The basic operations needed are:
 - Add an item to the top
 - Remove an item from the top
 - Check if the stack is full
 - Check if the stack is empty

Programming operations

- These methods could be written to implement the required functionality of a stack
 - `push(item)` – adds item to the top of the stack
 - `pop()` – removes and returns the item on the top of the stack
 - `isFull()` – checks if the stack is full
 - `isEmpty()` – checks if the stack is empty
- You might also want to write methods to:
 - `peek()` – return the top item without removing it
 - `size()` – return the number of items on the stack

Using a stack ADT

- In computing, a stack is an important data structure; one reason is that the order of insertion is the reverse of the order of removal
- Suppose you pop the letters **r**, **o**, **b**, **e**, **r**, **t** from a list **letters**, pushing each one onto a stack **s**
 - Now remove all these letters from the stack, adding each letter back into the list – what does the list look like now?

Algorithm for reversing a list

letters = ["r", "o", "b", "e", "r", "t"]

for each letter in letters

 remove letter from front of list

 push letter onto s

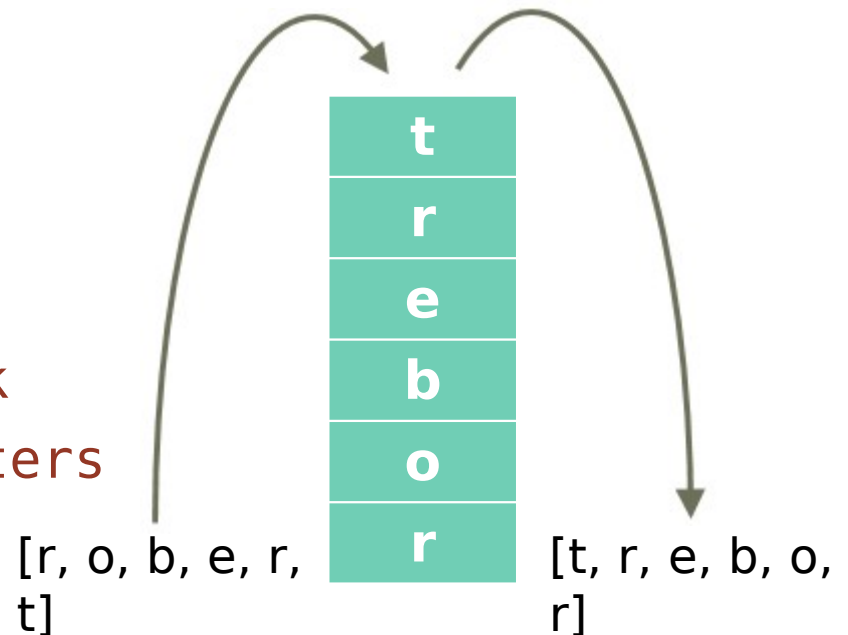
Next letter

For each letter in s

 pop letter from stack

 append letter to letters

Next letter



Implementing a stack as a list

- It's very easy to implement a stack as a list
- What list methods could you use to implement these functions?
 - Add an item to a stack
 - Remove an item from a stack
 - Check if the stack is empty
 - Find the number of items in the stack

Using a list

- Using built-in list operations, you could use the following list methods:
 - `append(item)` – add item to the top of the stack
 - `pop()` – remove and returns the item on the top of the stack
 - `len(stack)` – find the length (height) of the stack

Worksheet 4

- Complete **Task 1, questions 1 and 2** of the worksheet

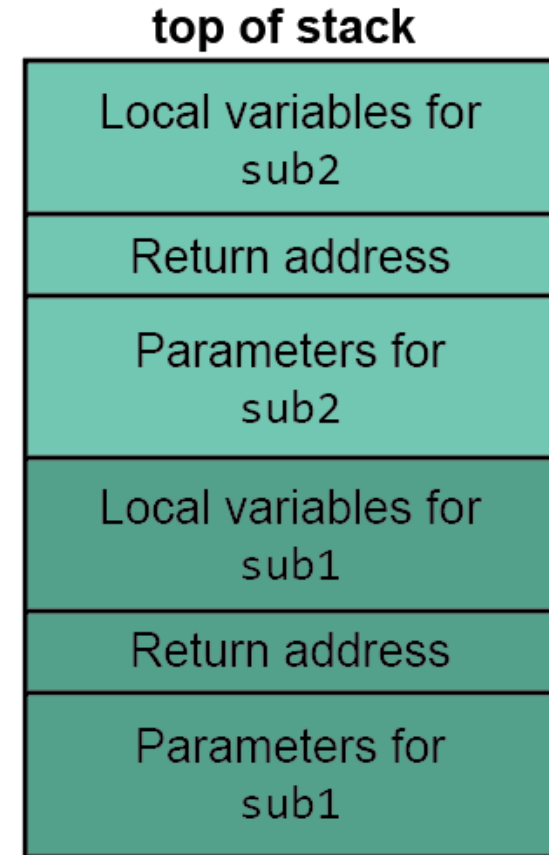


Overflow and underflow

- **Overflow** – attempting to push onto a stack that is full
- **Underflow** – attempting to pop from a stack that is empty
 - Note that if a stack is implemented using a dynamic list structure, there may be no “stack full” test. The computer may simply give a “stack overflow” error when it runs out of memory

Call stack

- The call stack is a **system level data structure**
- It provides the mechanism for passing **parameters** and **return addresses** to subroutines
- In high-level programming languages the use of the call stack is hidden from the programmer



Call stack

- What happens when this code is executed?

```
bigger = max (num1, num2)
```

- The programmer doesn't need to know how the arguments (`num1`, `num2`) are sent to the function `max`, or how the result is returned to the calling program
- The values of `num1` and `num2` and the `return address` (the line after the call statement), are saved on the stack
- These values are popped when the function completes

Subroutine calls

- Calls to subroutines are executed as follows:
 - The parameters are saved onto the stack
 - The address to which execution returns after the end of the subroutine is reached is saved onto the stack
 - Execution is transferred to the subroutine code

Subroutine execution

- Subroutines are executed as follows:
 - Stack space is allocated for local variables
 - The subroutine code executes
 - The return address is retrieved
 - The parameters are popped
 - Execution is transferred back to the return address

Plenary

- Describe a stack
- Describe the operations on a stack
- Give examples of the use of a stack
- Compare the behaviour of a stack with the behaviour of a queue

Plenary

- Stack - a last-in, first-out data structure
- Operations: PUSH and POP, test for full and empty stack
- Uses:
 - Holding return addresses, parameters and local variables when subroutines are called
 - Holding website addresses just visited
 - Holding operations just performed in word processor, spreadsheet etc.
- Compare with a queue – queue is FIFO, stack is LIFO

Copyright

© 2016 PG Online Limited

The contents of this unit are protected by copyright.

This unit and all the worksheets, PowerPoint presentations, teaching guides and other associated files distributed with it are supplied to you by PG Online Limited under licence and may be used and copied by you only in accordance with the terms of the licence. Except as expressly permitted by the licence, no part of the materials distributed with this unit may be used, reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic or otherwise, without the prior written permission of PG Online Limited.

Licence agreement

This is a legal agreement between you, the end user, and PG Online Limited. This unit and all the worksheets, PowerPoint presentations, teaching guides and other associated files distributed with it is licensed, not sold, to you by PG Online Limited for use under the terms of the licence.

The materials distributed with this unit may be freely copied and used by members of a single institution on a single site only. You are not permitted to share in any way any of the materials or part of the materials with any third party, including users on another site or individuals who are members of a separate institution. You acknowledge that the materials must remain with you, the licencing institution, and no part of the materials may be transferred to another institution. You also agree not to procure, authorise, encourage, facilitate or enable any third party to reproduce these materials in whole or in part without the prior permission of PG Online Limited.

